第4章

加速度センサ，角速度センサの活用事例

車輪移動型倒立振子の開発にみるセンサの使い方

熊谷正朗

加速度センサと角速度センサを搭載した倒立振子の設計事例を紹介する．傾斜角度そのものを直接把握できない角速度センサだけを搭載すると，センサや周辺回路のわずかな誤差の蓄積により，立ち続けることができないため，重力検知用に加速度センサを併用する．（編集部）

倒立振子は文字通り，「逆立ちした振り子」です．普通の振り子は**図1(a)**のように，上方に支持点があり，そこに重りがぶら下がっていて，ゆらゆら揺れるというイメージがあります．もう少し厳密に表せば，運動する物体(剛体)の重心が支持点の真下にあるときは動かず(平衡点[注1])，いずれかの方向に傾いた状態では，平衡点に戻ろうとする力(復元力)が働きます．これは重力によって引かれるためです．重りは摩擦などがなければいつまでも揺れていますが，普通は徐々に揺れが小さくなって(減衰)，最後には止まります．

その振り子を**図1(b)**のように逆立ちさせると(重心を支持点の上方に持って行くと)どうなるでしょうか．言うまでもなく，回転してぶら下がろうとします．

図1　振り子と逆立ちした振り子

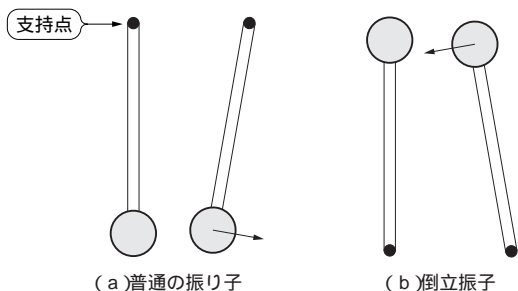
では，完全に重心を支持点の真上に持っていくとどうなるでしょうか．重りは支持点の真っすぐ上に乗っているため，左右に傾く力は働かず，動こうとはしません．ここも平衡点の一つです．ただし，ほんの少しでも傾くと，ますます倒す方向に力が働くため，あっという間に倒れてしまいます．

● ほうきが倒れないための制御方法

この，倒れるのが当たり前である逆立ちした振り子を，倒れないように立たせ続けることは可能です．小学校で掃除の時間にほうきを手の上に立てて遊んだ経験のある方も多いと思います．これがまさに倒立振子の状態です．倒れるようすを見ながら，それに合わせて手を動かすと，立たせ続けることができます．

この倒立振子は，昔から制御系に携わるエンジニアの関心事でした．やり方によってはできることは分かる，では「具体的にどのように」という問題の明確さや，一般に入力が一つで制御すべき値が4種類という手ごろな複雑さから，新しい制御理論はまず，倒立振子で試すという話もあります(ファジー制御で倒立振子など)．

● 倒立振子の考え方はロボットの足にも応用されている

その一方で近年は，移動ロボットの足として注目を集めています．乗り物として有名なセグウェイ(Segway)も，

注1：先の平衡点は安定平衡点，この平衡点は不安定平衡点．

KeyWord　加速度センサ，角速度センサ，平衡点，ステッピング・モータ，H8/3052，ADXRS150，ADXL202

人間を振り子の一部とする倒立振子です．また日立製作所のロボットのEMIEWも2車輪で動きます．何もしないと倒れてしまうということは危険に見えますが，常にバランスを取ることで思いがけない外部からの力に対応でき，また，機動性を向上させることができます．

そのほか，目に見えにくいのですが，歩行ロボットもその姿勢制御に倒立振子的な考えを入れている物があります．村田製作所のムラタセイサク君2号も，反動車型と呼ばれる倒立振子に分類されます．このような倒立振子においては，当然ですが，その傾きをいかに知るかが重要です[注2]．本記事では，**図1**に示す倒立振子の解説を通して，センサ・デバイスの選定，信号の扱い，ソフトウェア実装について述べます．

1. 製作する倒立振子の仕様を決める

今回紹介する倒立振子は，**写真1**に示すような機体で，筆者がかかわった倒立振子の中で6台目に当たります[注3]．この倒立振子の開発動機は，卒業研究をする学生が「玉乗りロボットを作りたい」と言い出したことです．いきなり玉は無理だからと説得して[注4]，「パイプ乗りロボット」にしました．その上で，卒業研究の範囲で手軽にまとめるために，以下のようなコンセプトとしました．

● 駆動はステッピング・モータで行う

世間一般には，DCサーボ・モータ＋減速機をトルク指令で動かしますが，さまざまな手間が増えます．そのため，扱いやすいステッピング・モータの軸に車輪を取り付け，直接，駆動しました．ついでにモータは左右独立にして，旋回などを可能にしました．

● 電池やコンピュータを搭載した自立型であること

倒立振子に限らず，実験室で装置を作るときは，電源を外に置き，制御もパソコンで，という形が手っ取り早いでしょう．ただし，今回の倒立振子の場合は，電線を引っ張ると制御にその影響が出やすいこと，パイプ乗りという移動を前提にした物であることから，電源もコンピュータも搭載した自立型にしました[注5]．

前者は，あくまで「手抜き」のために決めたことですが，これが想定外に極めて高い安定性をもたらしました．後者についても，いきなりデモしたりするのにとても便利な出来となり，「バランス・ロボット」として，これまでの作品の中で圧倒的な稼働率を誇っています．なお，**写真1**はパイプ乗りのための機能ははずした，単品で床に立つ機体です．

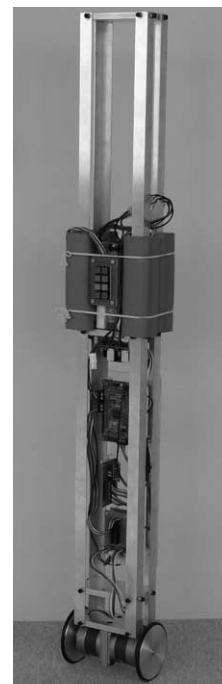



写真1
加速度センサと角速度センサを搭載した倒立振子

2. 倒れないように制御する方法

倒れる物を立てるためには，引き倒そうとする重力以上に，逆向きに作用する力を与え，それを調整する必要があります．本機の場合は，車輪で機体の下端を左右に加速することで，上が倒れてくるよりも速く，下端を動かして立て直します[注6]．

● 考慮すべき四つの状態がある

図2に倒立振子の制御において考慮する四つの状態を示します．**(0)**は目標とする状態です．機体は水平方向の目

注2：ムラタセイサク君もその本業は村田製作所のセンサの広報と推測される．
注3：倒立振子マニアというわけではないが，気が付いたら研究用や学生実験用などで7台．今年さらに2台追加の予定…．
注4：今回の車輪型は1方向にしか倒れないが，玉だと2方向に倒れるほか，ねじれも生じる．開発事例もほとんど見られない．
注5：持ち運んで簡単にデモができるように，という隠れた目的もあった．メカトロニクスの講義や高校での出張講義などにおいて，「種も仕掛けもありますが」と前置きしてから立たせると，それなりに関心を引ける．
注6：厳密には全体を加速することで生じる慣性力により，重力による転倒のモーメントを打ち消したり，さらに立て直したりする．

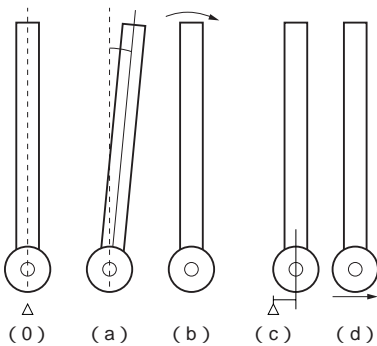
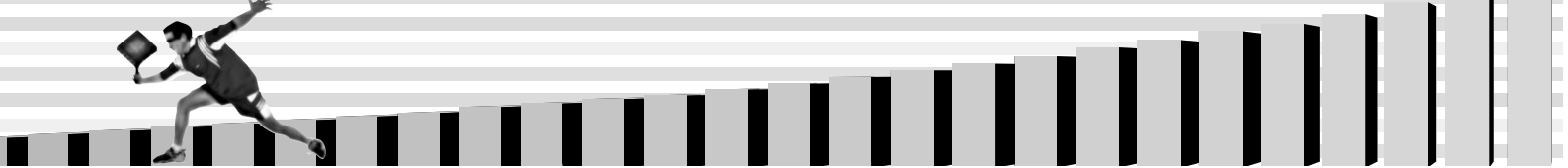



図2　倒立振子の制御において考慮する四つの状態
(a)は単純に右に傾いている．(b)は右に傾く速度が付いている．(c)は目標位置から右にずれている．(d)は右に動きつつある．

標位置△のところで，鉛直に立っています．この状態を崩すのは，(a)～(d)の四つの状態です．(a)は単純に右に傾いています．(b)は右に傾く速度が付いています．(c)は目標位置から右にずれています．(d)は右に動きつつあります．

この四つの状態を補正するには，おのおのどうすればよいでしょうか．答えはすべて，車輪で全体を「右に加速する」です．(a)，(b)は直感的に理解しやすいと思います．問題は，(c)，(d)です．右にずれている，ずれようとしているのに，さらに右に加速という矛盾があります．

実例を考えてみます．(0)の状態から，そのまま傾いて(a)になったとします．補正するためには，右に加速します．すると，全体的に右にずれますが，角度補正は瞬時にできるわけではないので，右に傾き，右にずれた状態になります．その状態で，右にずれたからと，左に加速すると，ますます右に倒れそうです．よって，左に加速するわけにはいきません．また，(c)，(d)のケースを無視すると，立てることだけに気を取られ，場合によってはどんどん加速しながら走っていきます．掃除の時間のほうき立てにおいて，上手に立て続けられる人のほかに，走れるだけ走っていってしまう人がいますが，まさにこの状態です．

消去法では，右に加速するしかなさそうです．右に大きく加速すると，下端が上端を追い抜き，結果的に左に倒れることになります．すると，この左に倒れた振り子を立て直すためには，左に加速しなければなりません．それが主になったとき，全体的に左に加速することになり，元の場所に戻ってくるわけです．これは直感的な現象の説明ですが，制御理論的にも「右に加速」で安定性が確認されています．この動作を繰り返すため，残念ながら(0)の状態で静止して立つことはできません．真っすぐ立っているようでもわずかに揺れています．

● 制御する四つのゲイン

以上をもとにした制御の式の一例は，傾斜角度θ，倒れる速度(角速度)ω，位置x，速度vから，出力すべき加速度aを算出すると，

$$a = K_\theta \theta + K_\omega \omega + K_x x + K_v v \quad \cdots\cdots(1)$$

となります．ここで，K_θ，K_ω，K_x，K_vはどのくらい反映させるかを決定する倍率定数(ゲイン)です．この四つのゲインを適切に調整することで立たせることができます．

通常，倒立振子の制御では，加速度ではなく，モータの出力すべきトルク(または水平推力)を求めます．条件にもよりますが，結果にはさほど違いがないこと[注7]，トルクを調整するには電流を調整できる回路と相応に特性の良いモータが必要なのに対して，目標加速度を積分すれば目標速度になって速度の制御はしやすいことなどから，この式を愛用しています．特に今回はステッピング・モータを使うので，トルクの式は使えません[注8]．

3．加速度センサと角速度センサを使う理由

● 姿勢の検出に求められる性能

式(1)に必要な，位置と速度は自分でモータを回しているので，自分で分かるはずです．ステッピング・モータでなくとも，通常，モータには相応の回転センサを付けておきます．そこで，倒立振子ではいかに姿勢角を測るかが重要になります．位置の誤差に比べ，姿勢の検出誤差は鉛直が狂うことであり，制御にとって致命的です．倒立振子に求められる姿勢センサの性能は，

- 十分な精度を持つこと．十分な分解能を持つこと(粗いと動きが粗くなる，落ち着かなくなる)．
- 十分な安定性を持つこと，つまり，長時間たってもゼロはゼロのままのこと(ゼロが狂えば倒れる)．
- 十分な応答性を持つこと(一般に，制御対象が小さいほど高い周波数まで必要．個人的目安は50Hz程度)．
- 移動の加速度など，姿勢以外の事象から影響を受けない

注7：高校で習う運動方程式では，$f = ma$と，力と加速度は比例している．もちろんそこまで単純化はできない．
注8：逆に加速度の式に慣れていたため，ステッピング・モータで作ろうなどと考えた．一般的には直流サーボ・モータなどを使う．

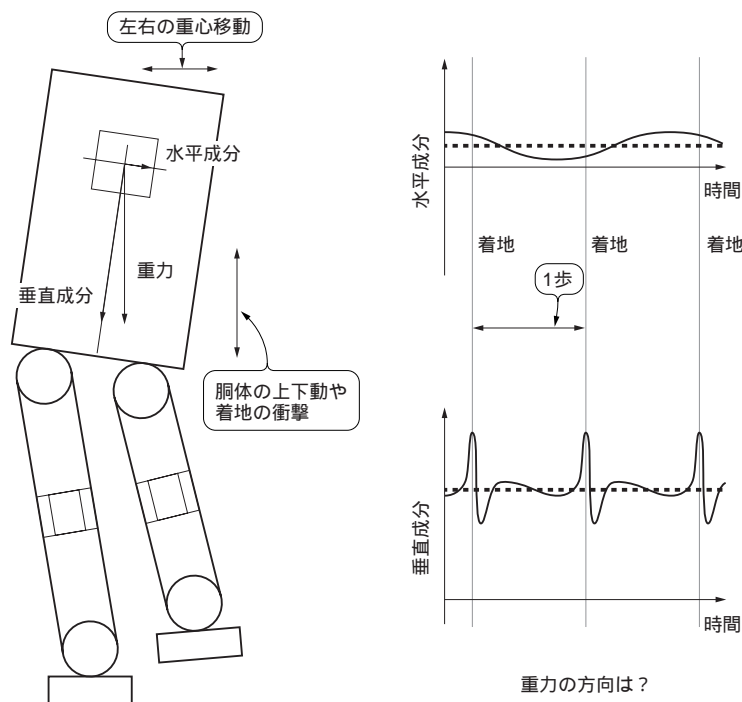



図3
2足歩行ロボットに加速度センサが搭載されたときの出力例
機体の傾きを加速度センサで把握するとき，センサ自体が揺すられると不必要な動きの加速度も検出してしまう．

こと．
などが挙げられます．加えて安ければいうことなしです．

● **姿勢検出の条件を満たすセンサ**

姿勢センサとしてよく知られている物に，角速度センサ（レート・ジャイロ）があります[注9]．カメラの手ぶれ補正や[注10]，自動車の姿勢検出などに使われていますし，歩行ロボットにも搭載されています．物にもよりますが，比較的多くのセンサが前項の特性を満たしています．

一つだけ注意したい点があります．それは検出する対象が傾斜角度そのものではなく，角速度であるという点です．角度を得るためには時間で積分しなければなりません．そのため，例えば温度の変化などで，静止状態のセンサの出力がわずかに0.1 °/s（1時間で1回転するほどの速度）程度ずれた場合でも，100sたてば10 °のずれとなってしまいます．10 °もずれれば，もはや角度のセンサとして使い物になりません．もちろん，センサ以外の増幅回路やA-Dコンバータの誤差でも同様です．

● **角度そのものを検出できるセンサも併用する**

そこで，通常は角度そのものを検出できるセンサを併用します．以前主流だったのは，傾斜計と呼ばれる，振り子に回転角度センサを付けたような物です．振り子が重力方向を向くため，その角度を測ることで傾斜角を測れます．

最近は性能が向上してきた加速度センサを使うようになりました．地球上には至る所に鉛直方向に「重力加速度」という加速度が働いています．そこで，これを加速度センサで検出すれば，重力の方向が分かります．しかし，これらの重力方向を検出するセンサにも弱点があります．センサ自体が揺すられると，その往復の加速度も検出してしまうという点です．倒立振子も左右に動きますし，歩行ロボットに至っては上下左右に激しく揺れます．

例えば，**図3**のようにロボットが一定の角度で胴体を傾けているとき，胴体に内蔵した加速度センサの出力は，重力だけならグラフの破線のような一定値となるはずですが，実際にはロボットが歩くときの左右の重心移動や，歩行に伴う上下動，着地衝撃などで実線のように歩行周期にあった変動が大きく現れ，この値だけで重力の方向は推定できません．そのため，平均値によって長時間にわたる絶対的な方向を知ることができても，短期間的には役に立たない結果が出てきます．

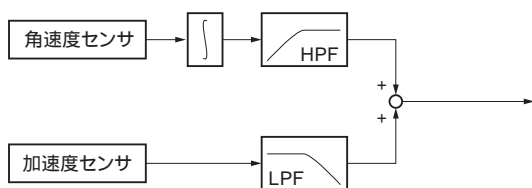
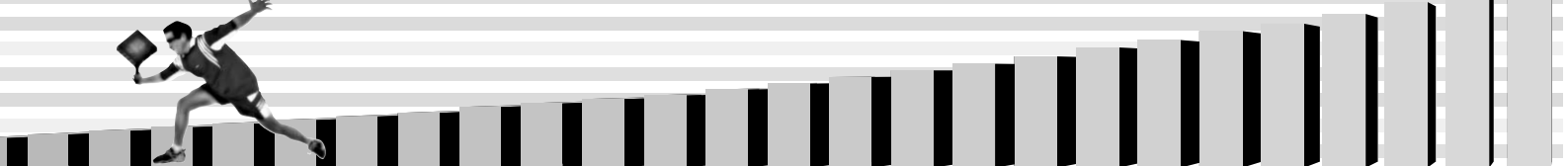
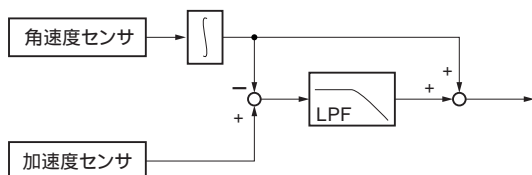
注9：角速度センサというとコマが回る機械的なセンサのイメージがあるが，単に姿勢のセンサくらいの意味合いで，普通は回る部品は内蔵していない．むしろ，圧電素子やMEMS部品を振動させる品種が主流．
注10：手ぶれはカメラの前後左右の揺れよりも，傾く方向の揺れのほうが像のぶれは大きくなる．

(a) フィルタを2個用いる場合

(b) フィルタを1個にした場合

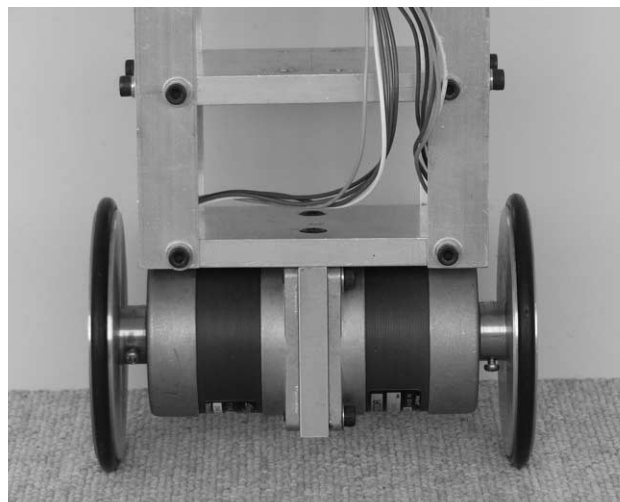
図4　加速度センサと角速度センサの出力をフィルタで合成

写真2　倒立振子の足はステッピング・モータ2個で構成

そこで，この二つを組み合わせることを考えます．角速度センサは短時間の性能に優れ，加速度センサは長時間の性能に優れます．これを周波数で考えると，角速度センサは周波数が高い方で，加速度センサは周波数の低い方で優れる，と言い換えられます注11．そこで，角速度センサの信号からハイパス・フィルタ（HPF）で欲しい成分を，加速度計からローパス・フィルタ（LPF）で欲しい成分を抜き取ってから混ぜればよいと考えられます．

図4(a)はその考えを図示した物です．角速度センサの信号はあらかじめ積分しておいて，角度同士で混合します．ここで，両センサが完ぺきなセンサだと仮定すると，フィルタを通して混ぜた後に特性が変化しないようにするためには，フィルタの特性として，LPF＋HPF＝1という関係が要求されます注12．

すると，HPF＝1－LPFと書き直せ，必要なHPFをLPFで実現できることになります．さらに，2系統でLPFを共有するようにすると，**図4(b)**のようにフィルタを1個で済ませることができます．アナログ回路で作る場合は部品が半分，ディジタル・フィルタで実装する場合も演算コストがほぼ半分になります．これを減算型フィルタ[(1)]と呼んでいます注13．

注11：一般的な電気の世界の発想では，どちらもかなりの低周波数．あくまで相対的な物で，境目は1Hzとか0.1Hzなど．
注12：厳密には，両者の伝達関数の和が1である．
注13：大学生時代にかかわってからずっと愛用している手法で，修士論文のテーマの半分はこれ．ただし，1軸単位で測定して混ぜる場合には使うが，3軸で大きく傾くときは適用できない．
注14：以前，東京・秋葉原の秋月電子で販売していたモータ．30個箱買いで1個当たり200円だった．
注15：重心が高い方が倒立振子を制御しやすいことは定説．

今回の倒立振子も，角速度センサと加速度センサ，減算型フィルタで姿勢検出を行っています．

4．倒立振子の足はステッピング・モータ2個で構成

メカ部分は，全体のきょう体となる部分と駆動部分からなります．移動の要である駆動部分を**写真2**に示します．ステッピング・モータ2個を背中合わせに1枚のアルミ板に固定し，全体の底板に当たるアルミ板に直角に固定しています．普通のステッピング・モータは軸が出ている側が固定面なのですが，たまたまこのモータ注14は軸の反対側が固定面で，この作りで精度良く軸が一致しました．モータには，直径約80mmのアルミ製ホイールにゴム製リングをはめてタイヤとした物を取り付けています．

きょう体部分は，アルミのアングルを4本用意し，底板および同じサイズのアルミ板3枚と組み合わせ，回路とバッテリの収納ができるようにしました（**写真1**）．この部分は制御にも影響しますが，経験的にパラメータ出しをする場合に厳密さは必要なく，「バッテリをそこそこ上の方に収納するように」注15とだけ学生に要望して，できたのがこの形です．

5．制御ハードウェアの開発

電子回路はきょう体内に格納しています．電源回路を除いて**写真3**のようにアクリル板に固定した上で，まとめて

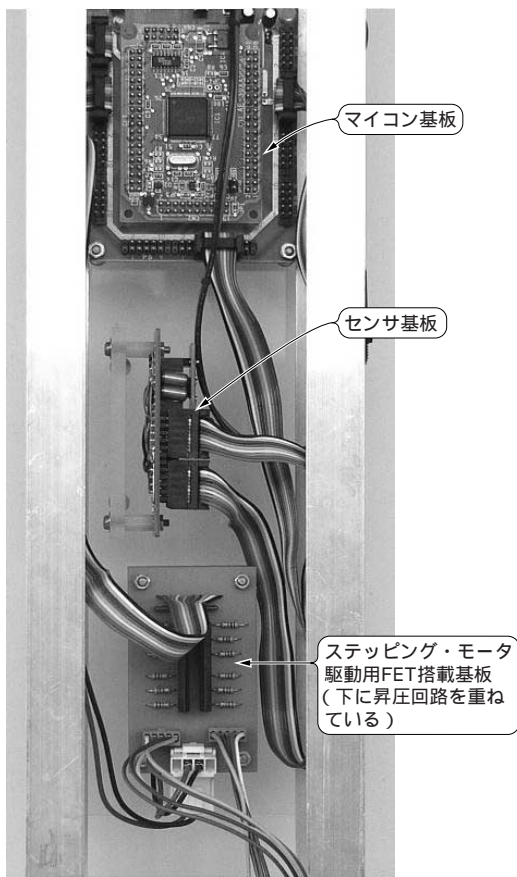



写真3　制御用マイコン搭載基板やセンサ，センサ周辺回路搭載基板の外観

（**a**）表面

（**b**）裏面

写真4　角速度センサ「ADXRS150」の外観

取り付けています．

● 制御用マイコン

制御にはルネサス テクノロジのマイコン「H8/3052」を使っています．必然性は特にありませんが，後述のセンサ類やモータ駆動回路を接続しうるI/O能力があり，なにより「使い慣れていた」ことが選定理由です．マイコンは秋月電子通商で販売されているマイコン・ボードの形で購入し，自作のポート分配基板（一種のコネクタ変換基板）に載せて使っています．

● センサの周辺回路

今回はセンサとして，角速度センサ（角速度を検出）と加速度センサ（重力加速度を検出）を使用しました．角速度センサには米国Analog Devices社のMEMS角速度センサ「ADXRS150」（**写真4**）を採用し，加速度センサは同社のMEMS加速度センサ「ADXL202」を採用しました．

角速度センサとしては，村田製作所，NECトーキンなどが製造している圧電素子系の品種が有名で，筆者自身も歩行ロボットの姿勢センサなどによく利用していました．たまたま，米国Analog Devices社のMEMS角速度センサのプレスリリースを知り，データシートを見てその性能に驚き，最近はもっぱら本品を使っています．それまで，歩行ロボットの研究などでカーナビ用や手ぶれ補正用の圧電系の角速度センサを（単体では不十分だったため）組み合わせて使っていましたが，それらの応答性（上限の周波数）や安定性（長時間のゼロ点出力の安定性），耐加速度性（衝撃などによるノイズ）の実測データと比較し，ADXRSのカタログ・スペックは凌駕していました．実際に使ってみても，確かに良好でした．

本機に搭載したセンサは，取り引き先経由で入手したサンプル品で，国内では普通の入手経路をまだ見つけていません．現在は米国の通販サイトSpark Fun Electrionics[注16]から，角速度センサおよび加速度センサを基板に実装済みの状態[注17]で，100ドル前後で購入しています．なお，ADXL202は秋月電子通商でも購入できます．

センサ周りの回路を**図5**に示します．上が角速度センサ，下が加速度センサの回路です．ADXRS150は2.5Vを基準に，角速度に比例した電圧が出力される仕様です．これはフルスケールが150°/sのセンサで，ADXRSシリーズでは感度が高いほうですが，実際に倒立振子を作ってみたところ，そのフルスケールほどの角速度は出ませんでした．そこで分解能重視にするため（マイコンのA-D変換器の分解能が10ビットしかない），単電源OPアンプで2.5Vを基準に，約2倍に増幅しています．

ADXRSの周りにコンデンサがいくつか付いていますが，これはデータシートのままとしました（内部で使う電源用のコンデンサなど）．なお，角速度はH8/3052のA-D変換

注16：http://www.sparkfun.com/
注17：ADXRSはBGA（ball grid array）パッケージのため，はんだ付けが困難．実装済みが便利．

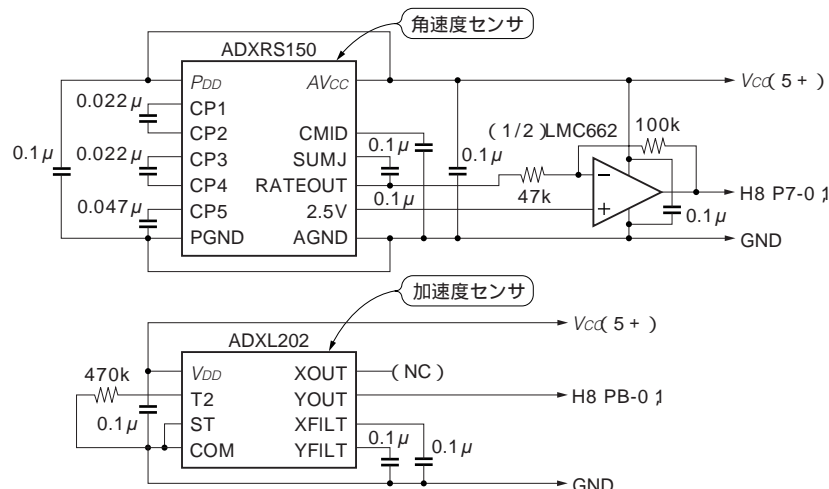



図5
センサ周りの回路
ADXL202はアナログ量のセンサとしては珍しく，検出値をパルスで出力する．

ポートの2本に入力しています．二つのポートの変換後の値を加えることで，おまじない程度に分解能の向上を期待しています．

ADXL202はアナログ量のセンサとしては珍しく，検出値をパルスで出力します．ほぼ一定周期のパルスを出力し，そのON時間の比率（デューティ比）が，加速度に比例して変わります．加速度ゼロでデューティ比が約50％です．これはこのまま，マイコンのパルス・カウンタに接続します．なお，計算を簡略化し，水平の加速度成分だけを利用しています．このセンサはX軸とY軸の計測が可能ですが，たまたまセンサをはんだ付けした向きの関係でY軸の出力だけをマイコンに送っています．

● ステッピング・モータ駆動回路

ステッピング・モータは，NチャネルFETを4個内蔵する東芝の「MP4401」で駆動しました（**図6**）．コイルの切り替え信号（励磁信号）は，マイコン側で作っています．今回のモータは定格電圧が24Vと高い一方，電流が280mAと少なめです[注18]．そこで，当初は7.2Vのニッカド電池3本（22V程度）で動かしていました[注19]．その後，**図7**の昇圧回路[注20]を開発し，追加しました．この回路は米国On Semi

注18：市販ステッピング・モータの多くは，コイルの定格電圧が数Vと低い代わりに電流が1A以上と多く，モータ2個を2相励磁するとすぐにトータルで5Aを超えたりして，電源設計が面倒．
注19：2000mAhの電池で2時間以上持つ．
注20：ロボットに取り付けるだけで性能が向上するため，実験室ではテム・レイ回路（笑）と呼ばれている．

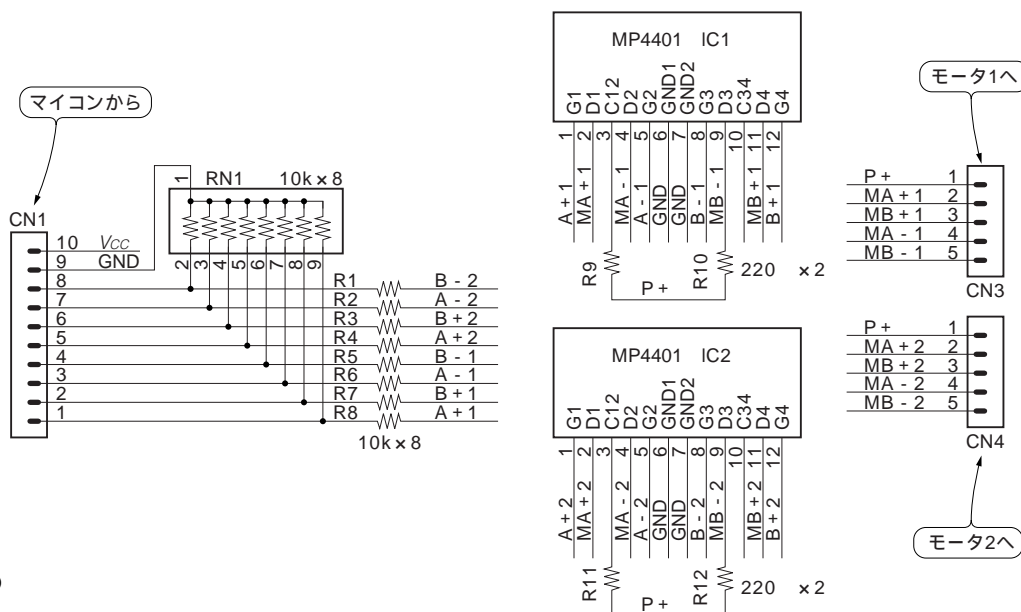



図6
ステッピング・モータ駆動回路
コイルの切り替え信号（励磁信号）は，マイコン側で作っている．

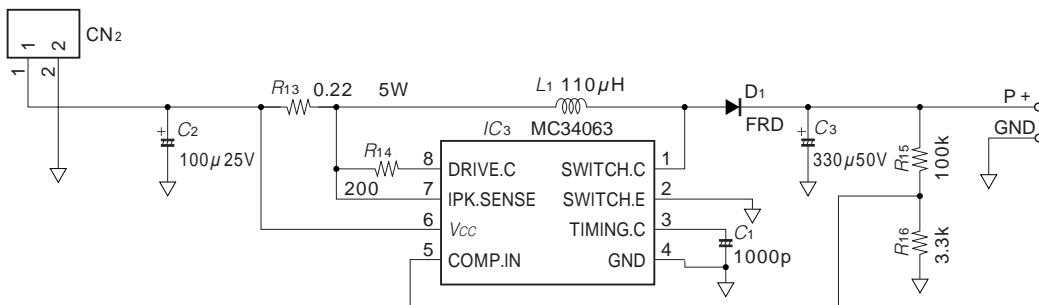



図7 モータを駆動するため用意した昇圧回路
無負荷時には約40Vが得られる．

conductor社製の「MC34063」を用いた昇圧回路で，無負荷時には約40Vが得られます．ただし，モータ停止状態(特定のコイルに直流電流が流れているとき)から低速回転時には，昇圧回路の入力電流制限によってほとんど昇圧されません．

一方，ステッピング・モータはその性質上，回転を速くするためにコイルの切り替え速度を高めると，電流が流れにくくなり，出力トルク(駆動力)が落ちるという難点があります．この回路は，その電流が流れにくい状態では可能な範囲で昇圧して，無理やり電流を流すようになります．結果的に，一種の定電力駆動のような特性が得られ，低速域で無駄に電流を消費することなく，高速域のトルクが向上しました．

ステッピング・モータでは，トルク不足による脱調が致命的な状況を起こしますが，この倒立振子の場合は即，転倒します．この回路によるトルク向上で，制御中につついたりしても倒れにくくなりました注21．

6．制御ソフトウェアの開発

以降，プログラムの主要部を，ソース・コードをもとに解説します．制御にかかわるすべての部分に共通する点として，固定小数点演算を多用していることが挙げられます．パワーのあるCPUであれば，浮動小数点演算回路が付いていたり，ソフトウェアでエミュレーションする余力もありますが，16ビットCPUで演算頻度を確保したい場合は採用できません．そこで，固定小数点を使います．

固定小数点は，例えば16ビットの変数に対して，上位8ビットを整数部，下位8ビットを小数部と(プログラマが)見なして，演算は通常の整数演算を行い，最後にけた合わせなどを行う手段です．別の見方をすれば，この場合は変数に2^8倍した値で格納しておくことになります．このとき加減算はそのまま，乗算は，

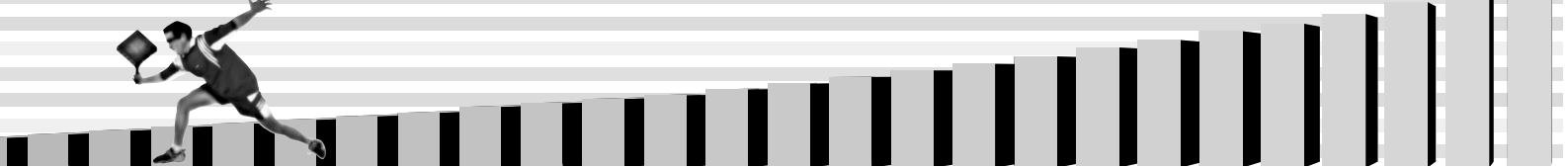
$$\left(A\times2^8\right)\times\left(B\times2^8\right)=AB\times2^{16}=\left(AB\times2^8\right)\times2^8 \quad \cdots\cdots(2)$$

となります．そこで，変数同士を乗じた後，2^8で割る，つまり，8ビット右シフト(>>8)を行えば，固定小数点での演算ができます．実際には，これをより強化(？)して，各値の最大の有効けた数をもとに，なるべく精度が残る(けた落ちしない)ように，小数点に当たる位置を決めて，各式でつじつまを合わせています．また，除算は演算に時間がかかるため，避けています．

● センサ信号処理

リスト1にセンサの信号処理部分の関数を示します．この関数は約8kHzのタイマ割り込み注22のたびに呼び出されます．この関数では，A-Dコンバータ経由で角速度センサ回路の出力電圧を(A-Dコンバータを2系統使うことでおまじない程度に分解能向上)，ITU(H8のタイマ)経由で加速度計の出力信号のON幅を取得しています．その後，変数gyroofscで大きく二つに分かれます．前半は通常の姿勢角度演算部，後半はセンサのオフセット計測部です．

まず，簡単なオフセット計測部を見てみます．オフセット計測とは，センサのオフセット，つまり，ゼロ点を検出することです．仕様上，角速度センサは角速度がゼロで2.5Vに，加速度計はデューティ比50％になります．しかし，各種誤差によるずれがあり，また温度変化によって徐々にずれてきます(温度ドリフト)．そのため，あらかじめゼロ点を決めておくことは難しく，電源投入時やずれて

注21：実際のところは，試しに昇圧回路を作ってみたら，たまたま良い特性が出たので，解析したらこうだった．その後量産され，実験室でロボットの性能強化に役立っている．
注22：H8でタイマ割り込みというと，もっぱら通常のタイマを使うが，タイマを他の目的で使いやすくするため，普段からウォッチドッグ・タイマのインターバル・タイマ・モードを使っている．

リスト1　センサの信号処理部分の関数

```
void AttitudeSensor(void)
{
  int gy,ac;
  long t;
  // 生値読み込み
  gy=(AD.DRA>>6)+(AD.DRB>>6);          // ジャイロ
  ac=ITU3.GRB;                          // 加速度センサ
  if(gyroofsc==0)                       // 通常演算モード
     {
        gyrov=(((long)gy)<<12)-gyroofs;
        gyroa+=(gyrov+0x80)>>8;
        acca=-((((long)ac)<<12)-accofs)*4;
        t=acca-gyroa;
        angflt=angflt+
                   (((t-angflt)+0x2000)>>14);
                   /* 0x2000-14 or 0x4000-15 */
        angle=gyroa+angflt;
     }
  else                             //オフセット除去モード
     {
        gyroofs+=gy;
        accofs+=ac;
        gyrov=0; gyroa=0; acca=0;
        angflt=0; angle=0;
        gyroofsc--;
     }
}
```

きたときに直接測定してしまうほうが現実的です．

具体的には，ゼロである状態(本機の場合は鉛直静止状態)で値を計測すればよく，また，ノイズの影響などを避けるためにある程度の回数の測定を行って平均値を得ます．このプログラムの場合は，メイン・ルーチン側で，角速度センサと加速度センサおのおののオフセット変数gyroofsとaccofsをゼロにクリアした上で，オフセット計測回数カウンタgyroofscを4096＝2^{12}にセットすることで，オフセット計測が開始します．後半部を4096回(約0.5s)通ると，gyroofscがゼロになり，gyorofs，accofsにはおのおののゼロ点平均値の2^{12}倍が入ります．併せて制御に使う角度変数群をゼロにクリアしています[注23]．

前半は角度演算部分です．まず，角速度センサに関する演算は，角速度センサの検出角速度ω_Gを積分して角度θ_Gを得ます．θ_Gは，

$$\theta_G = \int \omega_G dt \quad \cdots\cdots(3)$$

ですが，サンプリングされた値であるため，

$$\theta_G[i] = \sum_{k=0}^{i} \omega_G[k]\Delta T$$
$$\theta_G[i] = \theta_G[i-1] + \omega_G[i]\Delta T \quad \cdots\cdots(4)$$

と置き換えられます．厳密にSI単位系で計算を進めたい場合は，サンプリング周期ΔTは意味を持ちますが，ここでは，最終的にゲイン調整で済ませます(ゲインに含めてしまう)ので，気にせず省略します．まず，角速度値変数gyrovを，A-D変換から得た生の値から先ほどのオフセットを引くことで得ます．このとき，オフセット変数は平均値の2^{12}倍になっているので，生値のほうを2^{12}倍，つまり，12ビット左シフト(<<12)しています．いわば，小数点の位置が12ビット目にあります．オフセット計測回数を4096回と2^nの形にしていたのは，ここでシフト(一般に乗算に比べて高速)で済ませるためです．

次に角速度センサ角度gyroaに加え込んでいます．これが式(3)，式(4)を表しますが，加算の前に右に8ビット・シフトしています．これは今後のけたあふれを考慮して，適当に有効けた数を削減するためです．その際に，0x80を加えていますが，これは四捨五入に相当します．単に右シフトをすると切り捨てになりますが，シフト後に1の位に当たる数の半分を加えています[注24]．積分する際には，このちょっとの部分が大きく影響します．

同じように加速度センサによる角度取得を行います．まず，オフセットを引くところは同じです．本来は，鉛直方向および水平方向の検出加速度A_x，A_yをもとに，

$$\theta_A = \tan^{-1}\left(A_y / A_x\right) \quad \cdots\cdots(5)$$

で角度θ_Aを求めるべきです[注25]．しかし，本機の場合は傾斜角が小さいため，タンジェントの近似を使って，

$$\theta_A = \left(A_y / A_x\right) \quad \cdots\cdots(6)$$

さらに，傾斜が小さい場合は，A_xの変化は少ないため[注26]，適当な定数K_Aによって，

$$\theta_A = K_A A_y \quad \cdots\cdots(7)$$

と書けます．この形にするとK_Aにセンサやタイマの変換係数まで含めてしまうことができます．**リスト1**のaccaを求める式では，①冒頭のマイナスは角速度センサと加速度

注23：単純にメイン・ルーチンで4096回のループとせずに，わざわざ割り込み処理で行うのは，あえて間隔を置き，時間をかけて計測することで，ノイズなどの影響を抑えるもくろみがある．

注24：10進数を小数点以下を削って四捨五入する場合は，小数点以下第1位が5以上であれば繰り上げる，という操作だが，別の見方をすると，0.5＝1/2を加えて切り捨てるという操作である．普通のパソコンなどで演算する場合も，doubleをintにキャストするとき，ときどき0.5を加える．

注25：傾斜角度θのとき，A_x＝gcosθ，A_y＝gsinθ(gは重力加速度)となるため，A_y/A_x＝tanθとなる．

注26：A_x＝gcosθで，θが小さいうちはほとんどcosθ＝1と見なせるため．

センサの正負が不一致だったのでそれを解消，②「*4」は実験的に求めた角速度センサによる角度と加速度センサによる検出の比がたまたま4：1であったため，4倍して大きさを一致させた，という形にしています．

検出の比率は，gyroaとaccaを外部につないだパソコンからモニタしつつ，センサ回路を数度傾けてみて，値の変化から求めました．整数比にならなかった場合は，［*a)>>b］の形，つまり，($a/2^b$)倍することになります．今回の用途において加速度センサは鉛直であるかだけを見ていれば良いので，この程度の実装で十分機能しています．

その後は前述の角度合成の演算で，最終的に角度angle（θ）を求めます．数式表現は，

$$\theta = \theta_G + LPF(\theta_A - \theta_G) \quad \cdots\cdots(8)$$

です．このローパス・フィルタの実装は，1次のディジタル・フィルタにしました注27．式は，

$$y[k] = (1-r)y[k-1] + ru[k] \quad \cdots\cdots(9)$$

ただし，入力：$u[k]$，出力：$y[k]$，r：フィルタの特性を決める微小数値とする

です注28．ここで，$r = 1/2^n$と置き，書き換えると，

$$y[k] = y[k-1] + (u[k] - y[k-1]) / 2^n \quad \cdots\cdots(10)$$

となり，加減算とシフトだけの式になります．**リスト1**ではangfltが$y[k]$に当たり，$n = 14$で，ここでも四捨五入相当の演算を行っています．

● モータ駆動

ステッピング・モータは，ソフトウェアでマイクロステップ駆動を行っています．一般的には，ステッピング・モータでは各コイルの電流のONかOFFで回転させます．今回使ったようなコイルが4本あるタイプのモータの場合，順に1本ずつONすることで一定の角度ずつステップさせて回す1相励磁，2本ずつONする2相励磁，その両者を組み合わせ本来のステップの半分の角度ごとに回す1-2相励磁という3種類があります．

今回のモータの場合は，標準で1.8°ステップなので，1-2相励磁で0.9°になりますが，実際に動かしてみたところ，これでは粗い動きになりました注29．そこで，コイルに流す電流をON/OFFするだけではなく，中くらいの電流なども使うことで，疑似的に細かく励磁するマイクロステップという方法を使いました．普通は専用の駆動ICを使いますが，今回はONの時間比率を調整するPWM（pulse width modulation，加速度センサの出力と同じ）を用い，ソフトウェアで実装しました．具体的には，ON時間を(0/6)，(3/6)，(5/6)，(6/6)の比率としました（ON時間と電流は比例しないため）．

リスト2

```
#define SUBPAT 6
unsigned char STable[32*SUBPAT];
volatile unsigned int SPCr,SPCl;

void StepOut(void)
{
 unsigned char pl,pr;
 int stir,stil;
 SOseq=(SOseq+1)%SUBPAT;

 SPCr+=SSpeedr;
 SPCl+=SSpeedl;
 stir=(SPCr>>11)&0x1f;
 stil=(SPCl>>11)&0x1f;

 stir=stir+(SOseq<<5);
 stil=stil+(SOseq<<5);

 pr=STable[stir];
 pl=STable[stil];
 P1.DR.BYTE=pr|(pl<<4);
}
```

リスト2　ステッピング・モータ駆動部

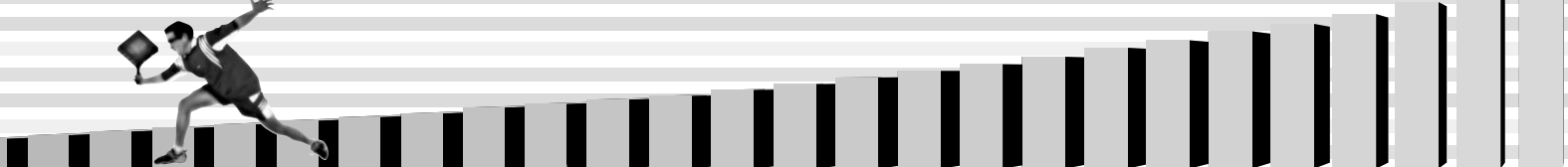
リスト2はモータ駆動の関数で，8kHz周期で呼ばれます．入力はメイン・ルーチンで指定される両輪の回転速度SSpeedr，SSpeedlです．ここで変数末尾のr，lは，モータの右左を意味します（以下，省略）．

まず，指定回転速度SSpeedより，励磁位相SPCを求めます．励磁位相($0 \sim 2^{16}-1$)は，4本のコイルの切り替え1セット分に当たります．位相の増加に伴って，励磁するコイルが順次切り替わり，オーバフロー(2^{16})で1本目のコイルに戻ります注30．ここでは，単純にSSpeedを加えています．

例えば，SSpeedが1ならば，65536回加算しないと一巡しませんが，SSpeedが4なら，16384回：「1/4の時間」：「4倍の速度」で1巡します．これは，10のような65536を割り切れない値を指定しても，問題なく動作します．

次に上位5ビットを切り出して，モータの励磁パターン

注27：1次だと甘いのだが，整数演算では2次以上は困難なため．
注28：極端な場合として，$r = 0$なら入力が反映されず，$r = 1$なら入力素通し．
注29：車輪が直径80mm程度であるため，1ステップで0.6mm程度動く．
注30：モータの仕様では4ステップ分に当たり，50回（＝200/4）あふれるとモータ1回転になる．

4

を得ています．5ビット（32通り）である理由は，1-2相励磁の8通りを基本として，そのON/OFFを4段階に分割したため，8×4＝32となっています．これにさらに，この関数が呼ばれるたびに0～5で変化する変数`SOseq`を上位に加え，その時点でのON/OFFを定めた数表`STable[]`のインデックス`sti`にしています．`sti`の下位を固定したまま上位を順に変えると，先ほどのON時間の比率に従ってON/OFFが決まるように`STable[]`を作っておきます．最後に，モータが接続されているI/Oポートから値を出力します．この際，8kHz/6＝1.3kHz周期でONになります．そのため，モータから，1.3kHzの音が「ピー」と小さく聞こえてきます．一般には耳障りとされていますが，ロボット開発者にとってはちょうど良い動作確認音です．

● **倒立振子制御**

倒立振子制御の本体は非常に簡単で，**リスト3**だけです（実際には，各種保護やテストのコードが数倍存在する）．`th`，`thv`，`x`，`v`はそれぞれ制御に必要な角度，角速度，位置，速度です．角度，角速度は姿勢センサの処理値を適宜シフトして使っています．速度は自分自身でモータを回しているのでその値を，位置はそれを積算して得ています．

倒立振子の制御式により，モータ加速度aを計算しています．`RegFileL[]`は，後述するパラメータ変数で，おのおのの制御ゲインにしています．式中で`>>14`しているように，これも固定小数値です．`thofs`，`xofs`は，制御開始を指示した時点での角度，位置で，ここを基準に（`th-thofs`をゼロにするように）制御されます．そのため，真っすぐ立てた状態で制御を開始します．求めた加速度を積算して目標速度`refv`にして，これをステッピング・モータへの指令速度にしています．左右逆向きにモータを付けているので，指令速度が正負逆です．これら，積算の式も，厳密にはサンプリング周期を乗じるべきですが，すべてゲインに含めてしまうことで，演算を削減しています．

リスト3　メイン・ルーチン内倒立振子制御部

```
th=angle>>12;
thv=gyrov>>8;
v=refv;
x=x+v;

a=
 ((RegFile[4]*(th-thofs))>>14)+
 ((RegFile[5]*thv))>>14)+
 ((RegFile[6]*(x-xofs))>>14)+
 ((RegFile[7]*v))>>14);

refv=refv+a;
SSpeedr= refv;
SSpeedr=-refv;
```

● **パラメータ通信設定**

本機に限らずマイコン制御でロボットなどを作る場合，内部の状態をモニタしたり，パラメータ調整をする手段は不可欠です．その際，特に制御物の場合，実行を一時停止するなど本来の制御に影響を及ぼすことはできず，バックグラウンドで入出力できる必要があります．その場合，簡易的にはディジタル入出力にLEDやスイッチを付けたり，A-Dコンバータを用いてパラメータを電圧で入れたり（可変抵抗器一つで済む），D-Aコンバータから変数を出力したり（連続値がオシロで確認できるという意味では非常に便利）する方法があります．ただし，分解能を確保したい場合や値の種類が多い場合は困難です．そこで，シリアル・ポート経由で，パソコンと通信するライブラリを作っています．詳細は触れませんが，

- シリアル通信割り込みだけで実装してあるため，本来の処理と完全に分離（限界の通信速度を出したときに，CPU時間の20%程度を使用）．
- マイコン側には16ビット・レジスタと32ビットのレジスタを各16本持ち，パソコン側から専用ソフトウェアで通信することでおのおのを読み書きできる．本来の処理の側からは，そのレジスタに代入したり，その値を使うのみ（強制的にパソコン側に値を送りつけることもできる）．
- 複数のマイコンを数珠つなぎにしたり，マイコン同士の通信も可能．

という仕様です．制御部の`RegFileL[]`は，この32ビット・レジスタで，パラメータ調整時にはパソコン側から値を適当に設定しては制御開始指令を出していました[注31]．

7．秘伝？倒立振子パラメータの調整法

さまざまな倒立振子を作り，パラメータを調整するうちに身につけた方法を参考までに紹介します．

注31：当初はセンサ初期化，制御開始などもすべてパソコン側から指示していた．現在は適切なパラメータも最初からマイコンに書き込んであり，パソコン抜きで動く．

1. すべてのゲインはゼロにしておく.
2. 手で倒立振子を支えつつ，角度のゲインを上げていく．少し倒すと起きあがる方向に動いてくるようになる.
3. 角度のゲインを上げると，動き始めたときに行きすぎて往復運動(発振)するようになる．ここで角速度のゲインを上げていくと収まるようになる.
4. 角速度のゲインで調整しつつ，角度のゲインを上げていく．すると，指先1本で振り子の上端を支えるだけで安定し，指で引っ張るとスルスルとついてくるようになる[注32]．静止した状態から，そっと手を離すと短時間は立ち，その後，つつーっと真っすぐ立ったまま走っていく(走らせすぎると当然倒れる).
5. 次に位置のゲインを上げていくと，指1本で引いたときは引いた方向と逆向きに多少倒れて抵抗するようになり，手を離した場合は，つつーっといって戻ってくるようになる．ただし，もとの場所を大きく通り過ぎて発振する.
6. 速度のゲインを上げていくと，発散していた位置がある振幅に収束するようになり，さらに上げるとその振幅が小さくなり，持続的に立つようになる.
7. 完全に静止直立にはならないが，角度・角速度のゲインと位置・速度のゲインを大きくすると揺れの振幅はある程度小さくでき(ゲインを上げすぎるとノイズなどに過敏になり悪化する)，それらの比率を調整すると姿勢と移動幅の振幅を調整できる.

ちなみに本機の場合，パラメータ調整を始めてから立つまで，5分かかりませんでした(個人的最短記録).

注32：お散歩モードと呼んでいる．これを応用して2輪の荷物運搬ロボットを試作した.
注33：水の入ったコップを上に乗せても怖くない程度．ステッピング・モータであるため，指定した通りに回る追従の良さと，間にギアなどを挟まないため，ガタによる制御の悪化が無いことが勝因と見ている.

＊　　＊　　＊

角速度センサ，加速度センサの応用例として，倒立振子の実例を紹介しました．本機の特徴は，角速度センサ，加速度センサを併用して姿勢角度の検出に用いていることと，ステッピング・モータ駆動であることです．センサの併用により，電池が切れるまで2時間でも一定の場所で立ち続けられる，姿勢検出の安定性を確保できました．また，ステッピング・モータのダイレクト・ドライブにより，ほとんど動かずに直立に近い制御を実現しました[注33]．ちなみに，おおもとの目的であったパイプ乗りは，この倒立振子にパイプからずり落ちないようにするためのローラを取り付けて行います．このとき，タイヤとパイプの摩擦により動力が伝達され，パイプが新たに駆動輪として振る舞うようになります．パラメータの再調整は必要ですが，安定して制御することができています．さらにその先の目標である玉乗りは，現在計画進行中ですが，ちょうど良い玉が見つからないのが目下の課題です….

参考・引用*文献

(1) 江村超，熊谷正朗，Lei WANG，郷古倫央；動歩行ロボット用レートジャイロのマルチセンサを用いた特性向上，計測自動制御学会論文集，Vol.35，No.1，pp.17-33，1999年.

くまがい・まさあき
東北学院大学工学部 機械知能工学科

<筆者プロフィール>

熊谷正朗．2000年，画像を含むセンサ情報による2脚歩行ロボットの斜面歩行に関する研究で東北大学にて工学博士を取得．同助手を経て，03年より東北学院大学講師，06年より助/准教授．主に，ロボットを作るのに必要な要素としてのセンシングの研究を行うほか，学生の希望に応じて，歩行ロボット，車輪ロボットなどさまざまなロボットの開発を技術面から支援することで，「ロボットをつくること」を多面的に検討．google://熊谷正朗

4

DESIGN WAVE MOOK　好評発売中
組み込みソフトウェア開発スタートアップ
ITエンジニアのための組み込み技術入門
Design Wave Magazine編集部 編　B5変型判 244ページ　定価2,310円(税込)　JAN9784789837194
CQ出版社　〒170-8461 東京都豊島区巣鴨1-14-2 販売部 ☎(03)5395-2141 振替 00100-7-10665